Colorado COVID-19: CDC API Analysis (Cache-First)

Table of contents

Inputs & cache strategy

This analysis reads pre-computed aggregates from `../cache/`.
If a file is missing, it will fetch the minimal slice from CDC and write it, so subsequent runs are fast and offline.

```
required_files <- c(
  "co_month.rds",
  "co_by_age.rds",
  # county×month: one file per year, we'll detect what's missing dynamically
  "co_county.rds",
  "severe.rds"
)

existing <- file.exists(file.path(CACHE_DIR, required_files))
tibble(file = required_files, exists = existing)
```

file	exists
co_month.rds	TRUE
co_by_age.rds	TRUE
co_county.rds	TRUE
severe.rds	TRUE

Build or load: Monthly totals for Colorado

```
if (!cache_exists("co_month.rds")) {
  require_token()
  message("Fetching co_month …")
  co_month <- cdc_select(
    select = "case_month, count(1) as n",
    where  = "res_state = 'CO'",
    group  = "case_month",
    order  = "case_month",
    limit  = 5000
  ) |>
    mutate(case_month = as.Date(paste0(case_month, "-01")),
           n = as.numeric(n))
  cache_write(co_month, "co_month.rds")
} else {
  co_month <- cache_read("co_month.rds")
}
summary(co_month)
```

```
   case_month                n          
 Min.   :2020-01-01   Min.   :    85  
 1st Qu.:2021-02-01   1st Qu.:  9304  
 Median :2022-03-01   Median : 17717  
 Mean   :2022-03-02   Mean   : 35542  
 3rd Qu.:2023-04-01   3rd Qu.: 43416  
 Max.   :2024-05-01   Max.   :329820  
```

Build or load: Monthly by age group

```r
if (!cache_exists("co_by_age.rds")) {
  require_token()
  message("Fetching co_by_age …")
  co_by_age <- cdc_select(
    select = "case_month, age_group, count(1) as n",
    where  = "res_state = 'CO' AND age_group IS NOT NULL AND age_group <> 'Missing'",
    group  = "case_month, age_group",
    order  = "case_month, age_group",
    limit  = 50000
  ) |>
    mutate(case_month = as.Date(paste0(case_month, "-01")),
           n = as.numeric(n))
  cache_write(co_by_age, "co_by_age.rds")
} else {
  co_by_age <- cache_read("co_by_age.rds")
}
summary(co_by_age)
```

```
   case_month          age_group                n
 Min.   :2020-01-01   Length:257         Min.   :    24
 1st Qu.:2021-03-01   Class :character   1st Qu.:   449
 Median :2022-04-01   Mode  :character   Median :  2455
 Mean   :2022-03-26                      Mean   :  7329
 3rd Qu.:2023-05-01                      3rd Qu.:  7563
 Max.   :2024-05-01                      Max.   :184940
```

County × Month slices (yearly) — load existing, fetch only missing

```r
# County × Month slices (yearly) - load existing, fetch only missing (robust to empty years)

# Use co_month to cap the year range to what's actually available
year_start <- 2020L
last_month <- max(co_month$case_month, na.rm = TRUE)
year_end   <- lubridate::year(last_month)
target_years <- seq.int(year_start, year_end)

year_files   <- sprintf("co_county_month_%d.rds", target_years)
missing_years <- target_years[!file.exists(file.path(CACHE_DIR, year_files))]

# Helper: fetch one year's slice; return a typed-empty tibble if API returns []
```

```r
fetch_county_month_year <- function(y) {
  df <- cdc_select(
    select = "case_month, county_fips_code, count(1) as n",
    where  = sprintf("res_state = 'CO' AND county_fips_code IS NOT NULL AND case_month betwe
    group  = "case_month, county_fips_code",
    limit  = 120000
  )

  # If the API returns zero rows or missing columns, create a typed-empty tibble
  if (nrow(df) == 0L || !all(c("case_month","county_fips_code","n") %in% names(df))) {
    message("No rows for year ", y, " - skipping (dataset likely ends before this year).")
    return(tibble::tibble(
      case_month       = as.Date(character()),
      county_fips_code = character(),
      n                = numeric()
    ))
  }

  df |>
    dplyr::mutate(
      case_month       = as.Date(paste0(case_month, "-01")),
      county_fips_code = sprintf("%05s", county_fips_code),
      n                = suppressWarnings(as.numeric(n))
    )
}

# Fetch only the missing years (if any)
if (length(missing_years)) {
  require_token()
  message("Fetching county×month slices for: ", paste(missing_years, collapse = ", "))
  for (y in missing_years) {
    df_y <- fetch_county_month_year(y)
    cache_write(df_y, sprintf("co_county_month_%d.rds", y))
  }
}

# Bind all available yearly slices from cache
yr_files <- list.files(CACHE_DIR, pattern = "^co_county_month_\\d{4}\\.rds$", full.names = T

if (length(yr_files) == 0L) {
  warning("No county-month cache files found after fetch; proceeding with an empty frame.")
  co_county_month <- tibble::tibble(
```

```
    case_month       = as.Date(character()),
    county_fips_code = character(),
    n                = numeric()
  )
} else {
  co_county_month <- purrr::map_dfr(yr_files, readRDS) |>
    dplyr::mutate(
      case_month       = as.Date(case_month),
      county_fips_code = sprintf("%05s", county_fips_code),
      n                = suppressWarnings(as.numeric(n))
    ) |>
    dplyr::arrange(case_month, county_fips_code)
}

summary(co_county_month)
```

```
   case_month         county_fips_code         n          
 Min.   :2020-01-01   Length:1326        Min.   :   11.00  
 1st Qu.:2021-03-01   Class :character   1st Qu.:   89.25  
 Median :2022-03-01   Mode  :character   Median :  342.00  
 Mean   :2022-03-19                      Mean   : 1420.59  
 3rd Qu.:2023-04-01                      3rd Qu.: 1303.50  
 Max.   :2024-05-01                      Max.   :45967.00  
```

County totals — derive locally from county × month, then join names

```
# Use cache if present; otherwise derive and save
if (cache_exists("co_county.rds")) {
  co_county <- cache_read("co_county.rds")
} else {
  co_county <- co_county_month |>
    group_by(county_fips_code) |>
    summarise(n = sum(n, na.rm = TRUE), .groups = "drop")

  suppressPackageStartupMessages({ library(sf); library(tigris) })
  options(tigris_use_cache = TRUE)
  co_names <- tigris::counties(state = "CO", year = 2023, class = "sf") |>
    sf::st_drop_geometry() |>
    transmute(county_fips_code = GEOID, res_county = NAME)
```

```
  co_county <- co_county |>
    left_join(co_names, by = "county_fips_code") |>
    relocate(res_county, .after = county_fips_code) |>
    arrange(desc(n))

  cache_write(co_county, "co_county.rds")
}
summary(co_county)
```

```
 county_fips_code    res_county              n
 Length:27          Length:27          Min.   :  5756
 Class :character   Class :character   1st Qu.:  8664
 Mode  :character   Mode  :character   Median : 19403
                                       Mean   : 69767
                                       3rd Qu.:110558
                                       Max.   :251809
```

Severity proxies (hospitalization & death ratios) — always present

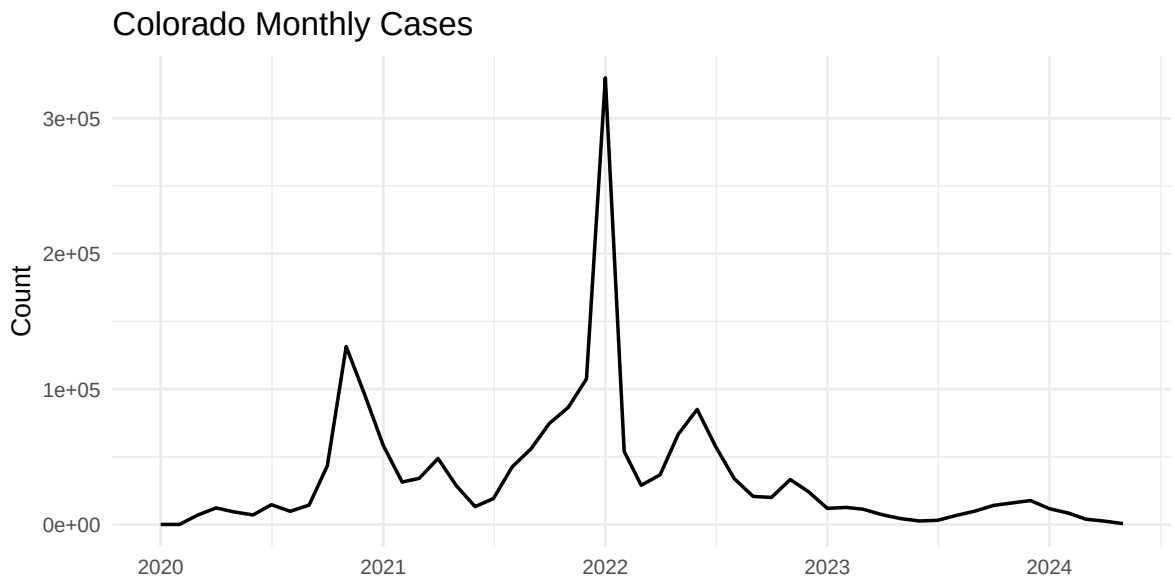
```
if (!cache_exists("severe.rds")) {
  require_token()
  message("Computing severity ratios (from grouped queries)…")

  count_by_month <- function(where_extra = NULL, limit = 50000) {
    wc <- paste(c("res_state = 'CO'", where_extra), collapse = " AND ")
    cdc_select(
      select = "case_month, count(1) as n",
      where  = wc,
      group  = "case_month",
      order  = "case_month",
      limit  = limit
    ) |>
      mutate(case_month = as.Date(paste0(case_month, "-01")),
             n = as.numeric(n))
  }

  co_total <- count_by_month()
  co_hosp  <- count_by_month("hosp_yn = 'Yes'") |> rename(hosp_n = n)
  co_death <- count_by_month("death_yn = 'Yes'") |> rename(death_n = n)
```

```
  severe <- co_total |>
    full_join(co_hosp,  by = "case_month") |>
    full_join(co_death, by = "case_month") |>
    mutate(
      hosp_n     = tidyr::replace_na(hosp_n, 0),
      death_n    = tidyr::replace_na(death_n, 0),
      hosp_rate  = if_else(n > 0, hosp_n / n, NA_real_),
      death_rate = if_else(n > 0, death_n / n, NA_real_)
    ) |>
    arrange(case_month)

  cache_write(severe, "severe.rds")
} else {
  severe <- cache_read("severe.rds")
}
summary(severe)
```

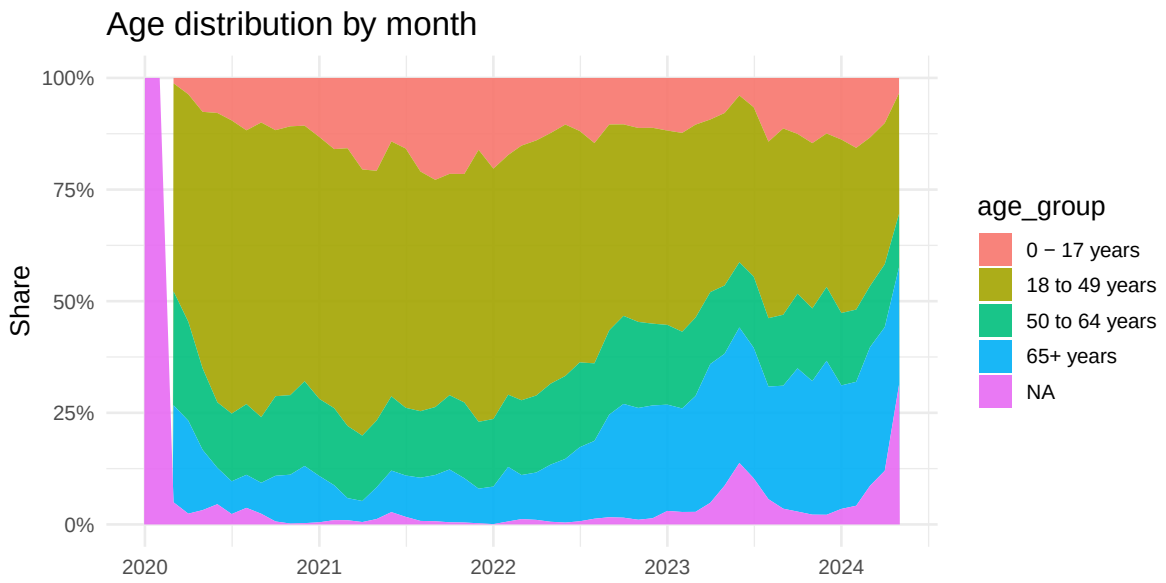
```
   case_month                 n                hosp_n          death_n
 Min.   :2020-01-01   Min.   :    85   Min.   :   4   Min.   :  0.00
 1st Qu.:2021-02-01   1st Qu.:  9304   1st Qu.: 572   1st Qu.:  0.00
 Median :2022-03-01   Median : 17717   Median :1084   Median :  0.00
 Mean   :2022-03-02   Mean   : 35542   Mean   :1602   Mean   : 87.53
 3rd Qu.:2023-04-01   3rd Qu.: 43416   3rd Qu.:2073   3rd Qu.: 24.00
 Max.   :2024-05-01   Max.   :329820   Max.   :6760   Max.   :950.00
   hosp_rate         death_rate
 Min.   :0.01171   Min.   :0.0000000
 1st Qu.:0.04684   1st Qu.:0.0000000
 Median :0.06250   Median :0.0000000
 Mean   :0.06765   Mean   :0.0022604
 3rd Qu.:0.07714   3rd Qu.:0.0009324
 Max.   :0.29444   Max.   :0.0444083
```

Quick visuals (sanity checks)

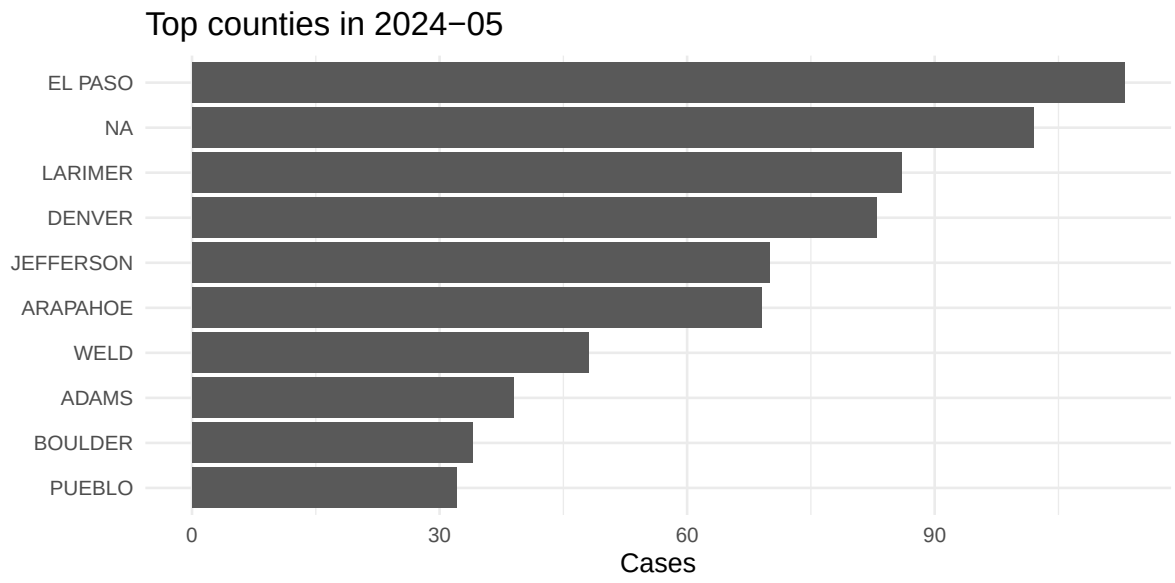
```
co_month |>
ggplot(aes(case_month, n)) +
  geom_line(linewidth = 0.8) +
  labs(title = "Colorado Monthly Cases", x = NULL, y = "Count") +
  theme_minimal(base_size = 13)
```




```
co_by_age |>
  group_by(case_month) |>
  mutate(pct = n / sum(n)) |>
  ungroup() |>
  ggplot(aes(case_month, pct, fill = age_group)) +
  geom_area(alpha = 0.9) +
  scale_y_continuous(labels = percent) +
  labs(title = "Age distribution by month", x = NULL, y = "Share") +
  theme_minimal(base_size = 13)
```




```
latest_month <- max(co_county_month$case_month, na.rm = TRUE)
co_county_month |>
  filter(case_month == latest_month) |>
  left_join(select(co_county, county_fips_code, res_county), by = "county_fips_code") |>
  slice_max(n, n = 10) |>
  mutate(res_county = forcats::fct_reorder(res_county, n)) |>
  ggplot(aes(res_county, n)) +
  geom_col() +
  coord_flip() +
  labs(title = paste("Top counties in", format(latest_month, "%Y-%m")),
       x = NULL, y = "Cases") +
  theme_minimal(base_size = 13)
```



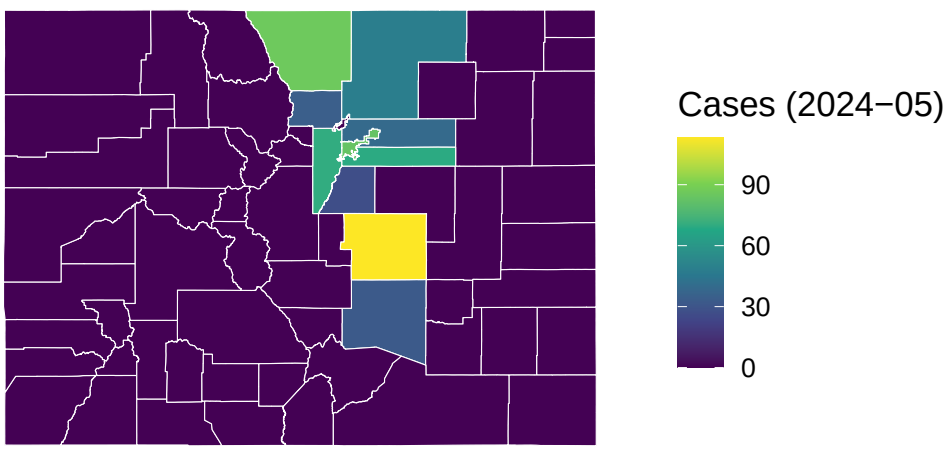

```
suppressPackageStartupMessages({ library(sf); library(tigris); library(dplyr); library(ggplo
```

```
Warning: package 'sf' was built under R version 4.4.1
```

```
Warning: package 'tigris' was built under R version 4.4.1
```

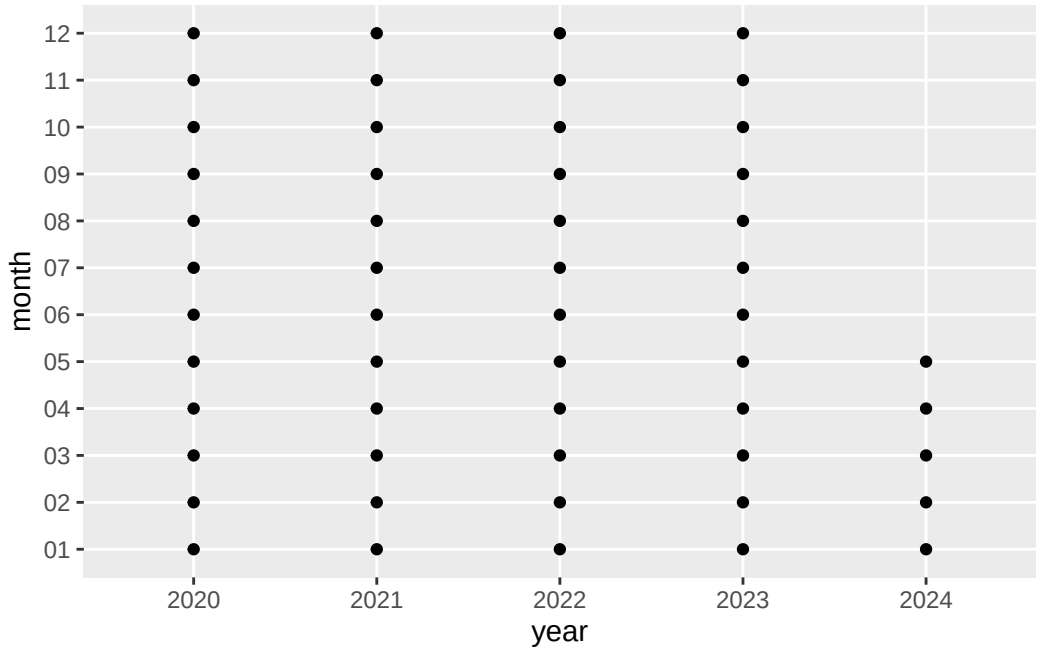
```
options(tigris_use_cache = TRUE)

# 1) Pick a month to map (latest available by default)
stopifnot(exists("co_county_month"), nrow(co_county_month) > 0)
map_month <- max(co_county_month$case_month, na.rm = TRUE)

# 2) Get Colorado county polygons (cached locally by tigris)
co_shapes <- tigris::counties(state = "CO", year = 2023, class = "sf") |>
  dplyr::transmute(county_fips_code = GEOID, geometry)

# 3) Aggregate counts for the selected month
df_month <- co_county_month |>
  dplyr::filter(case_month == map_month) |>
  dplyr::group_by(county_fips_code) |>
  dplyr::summarise(n = sum(as.numeric(n), na.rm = TRUE), .groups = "drop")

# 4) Join counts to shapes; missing counties get 0
map_df <- co_shapes |>
```

```
  dplyr::left_join(df_month, by = "county_fips_code") |>
  dplyr::mutate(n = dplyr::coalesce(n, 0))

# 5) Plot
ggplot(map_df) +
  geom_sf(aes(fill = n), color = "white", linewidth = 0.2) +
  scale_fill_viridis_c(labels = label_number(), name = paste0("Cases (", format(map_month, "%
  labs(
    title = "Colorado COVID-19 - County-level cases",
    subtitle = paste("Month:", format(map_month, "%B %Y")),
    caption = "Counts are monthly totals from CDC SODA (cached)."
  ) +
  theme_minimal(base_size = 13) +
  theme(
    axis.text = element_blank(),
    axis.title = element_blank(),
    panel.grid = element_blank(),
    legend.position = "right"
  )
```




```
severe |>
  mutate(case_month = ymd(case_month)) |>
```

```
  arrange(case_month) |>
  separate(case_month,
           into = c("year", "month", "day"),
           sep = "-",
           remove = FALSE) |>
  ggplot(aes(x = year, y = month)) +
  geom_point()
```

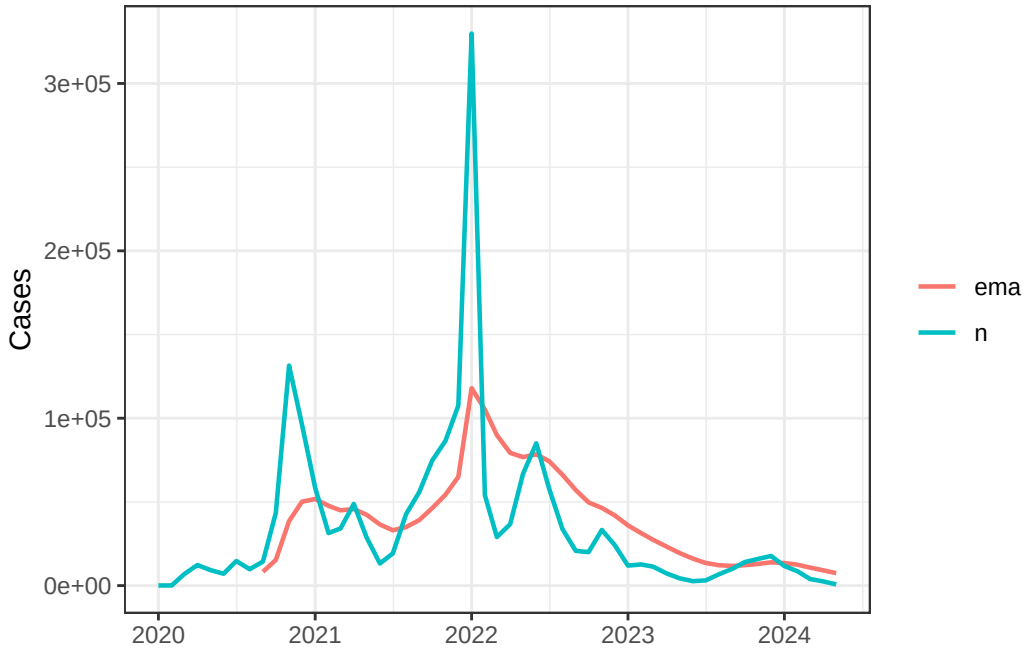



```
df <- severe |>
  mutate(case_month = ymd(case_month)) |>
  arrange(case_month) |>
  transmute(case_month, n = as.numeric(n)) |>
  mutate(ema = TTR::EMA(n, ratio = 0.2))

df |>
  select(case_month, n, ema) |>
  pivot_longer(cols = c("ema", "n"),
               names_to = "type",
               values_to = "value") |>
  ggplot(aes(x = case_month, y = value, group = type)) +
    geom_line(aes(color = type),
              size = 0.8) +
```

```
  labs(x = NULL,
       y = "Cases",
       color = NULL) +
  theme_bw()
```

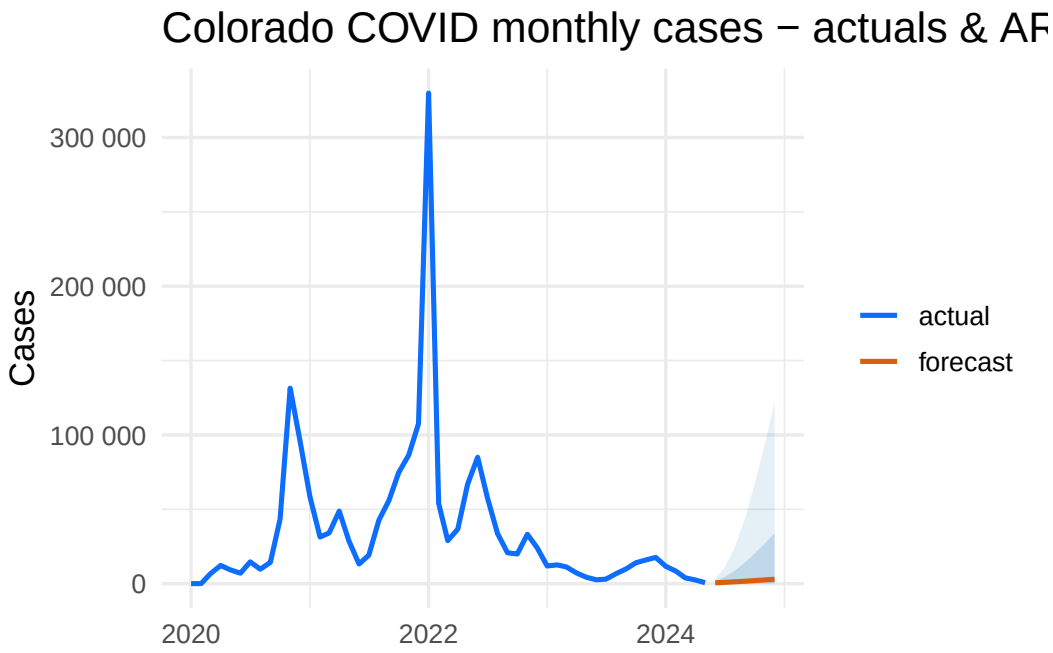
```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```

```
Warning: Removed 8 rows containing missing values or values outside the scale range
(`geom_line()`).
```




```
library(forecast)  # install.packages("forecast")

# make a monthly ts object (frequency=12)
start_year  <- lubridate::year(min(df$case_month, na.rm = TRUE))
start_month <- lubridate::month(min(df$case_month, na.rm = TRUE))
y_ts <- ts(df$n, start = c(start_year, start_month), frequency = 12)

# fit ARIMA on log1p scale
fit <- forecast::auto.arima(log1p(y_ts), stepwise = TRUE, approximation = FALSE)
```

```r
# horizon to Dec 2024
last_obs <- max(df$case_month, na.rm = TRUE)
h <- max(0, (12 * (2024 - lubridate::year(last_obs))) +
              (12 - lubridate::month(last_obs)))  # months until Dec 2024

fc <- forecast::forecast(fit, h = h)

# back-transform and build a tidy frame
future_dates <- seq(from = last_obs %m+% months(1), by = "month", length.out = h)
pred <- tibble::tibble(
  case_month = future_dates,
  n_hat = expm1(as.numeric(fc$mean)),
  lo80  = expm1(as.numeric(fc$lower[, "80%"])),
  hi80  = expm1(as.numeric(fc$upper[, "80%"])),
  lo95  = expm1(as.numeric(fc$lower[, "95%"])),
  hi95  = expm1(as.numeric(fc$upper[, "95%"]))
)

# combine actuals + forecast for plotting
plot_df <- dplyr::bind_rows(
  df %>% transmute(case_month, value = n, type = "actual"),
  pred %>% transmute(case_month, value = n_hat, type = "forecast")
)

ggplot() +
  geom_ribbon(data = pred, aes(case_month, ymin = lo80, ymax = hi80), alpha = 0.2, fill = "#
  geom_ribbon(data = pred, aes(case_month, ymin = lo95, ymax = hi95), alpha = 0.12, fill = "
  geom_line(data = plot_df, aes(case_month, value, color = type), linewidth = 0.9) +
  scale_color_manual(values = c(actual = "#0d6efd", forecast = "#d95f0e")) +
  scale_y_continuous(labels = scales::label_number()) +
  labs(x = NULL, y = "Cases", color = NULL,
       title = "Colorado COVID monthly cases - actuals & ARIMA forecast to Dec 2024") +
  theme_minimal(base_size = 13)
```


Colorado COVID monthly cases − actuals & AF
Cases
300 000
200 000
100 000
0
2020
2022
2024
actual
forecast